STEP IT ACADEMY

Programming with python

Lesson # 2

Python Turtle Graphics. Simple Figures

CONTENTS

Introduction

Python has a cool option of drawing with a turtle.

You can write commands and indicate where it should go: forward, backward, left, right. By default, it is represented by an object that looks like an arrow. This object can be turned into a turtle with just one line of code, and now we'll learn how to do it.

Let's get started!

Your First Program

Each geometric figure can consist of points, lines, and curves, for example, a square, a triangle, a circle, a rectangle, or a polygon. In this lesson we will learn how to draw all these elements using an object from the **turtle** library. You can see this module in the documentation at the link.

Python has lots of open libraries, and it's impossible to remember what each of them is designed for, so do not forget to refer to the official documentation. There are the most frequently used modules and we will work with them later. At the moment, we are interested in **turtle**.

First, create a new window and display the turtle in it (Figure 1):

```python
import turtle

window = turtle.Screen()
turtle.Pen()
turtle.shape("turtle")
window.exitonclick()
```

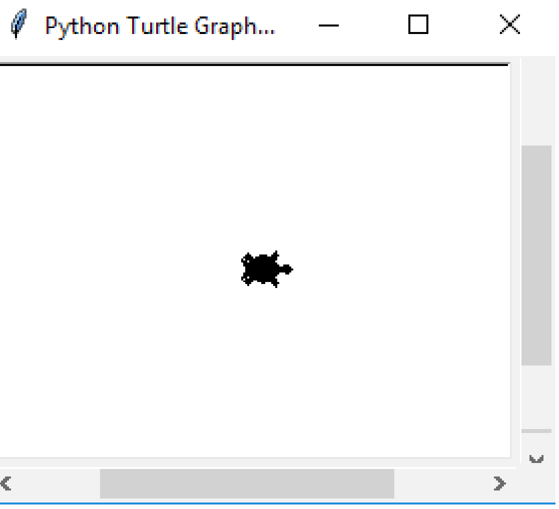

Figure 1

In the code, we imported **turtle** (`import turtle`) so that we could work with this module in the future. Then we created a new window `turtle.Screen()`so that all the actions we performed were displayed in it. In the window, we displayed the `turtle.Pen()` pointer and set the `turtle.shape("turtle")` style.

You can choose the appearance you like by simply changing the value of `"turtle"` to one of the suggested ones: `"arrow"`, `"circle"`, `"circle"`, `"triangle"`, `"classic"`. Try changing the style and select the one that you like the most.

Also we added the line `window.exitonclick()` so that the new window closes after we left-click on it.

Drawing a Rectangle

Our turtle can move and create different figures!

The process of drawing is very simple: the object moves and leaves a trace behind it. Imagine that the turtle is at the point `0`, this is intersection of the `x` and `y` axes and our center (Figure 2).

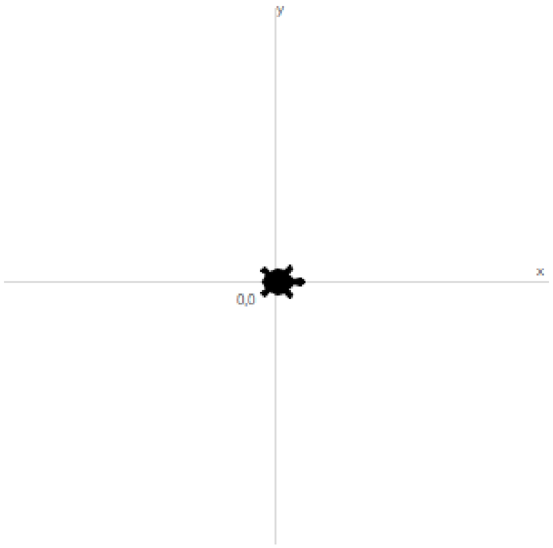

Figure 2

Let's teach our turtle to draw a rectangle (see Figure 3 on page 7):

```python
import turtle

window = turtle.Screen()
turtle.reset()

turtle.shape("turtle")
turtle.bgcolor("red")
turtle.color("white")

turtle.speed(2)
turtle.pensize(3)

turtle.forward(150)
turtle.left(90)
turtle.forward(75)
turtle.left(90)
turtle.forward(150)
turtle.left(90)
turtle.forward(75)

window.exitonclick()
```

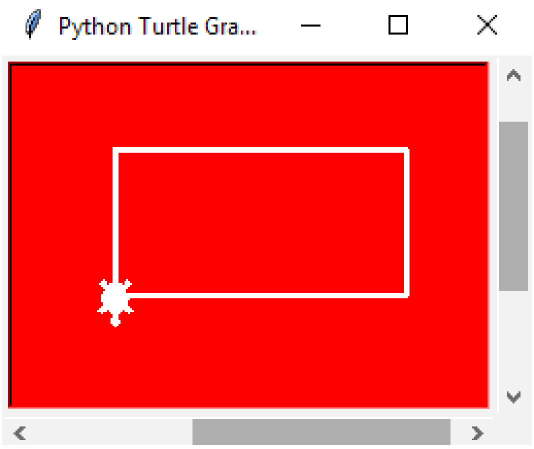

Figure 3

Let's see what our code does.

The first thing we did was to clear the canvas before drawing `turtle.reset()`. Also here we added the background color `turtle.bgcolor("red")` and set the color of the lines `turtle.color("white")`.

To set the turtle speed, we wrote the function `turtle.speed(2)`. The smaller the number in parentheses, the slower the object will move. Try to change this value to `0`, `5`, `10` and look at the result.

To make the line not too thin, we set the size (thickness) equal to three: `turtle.pensize(3)`. You can also change this parameter.

So, **a rectangle is a figure in which two sides are pairwise equal**, i. e. AB=CD and BC=DA, and **all angles are straight** (Figure 4).

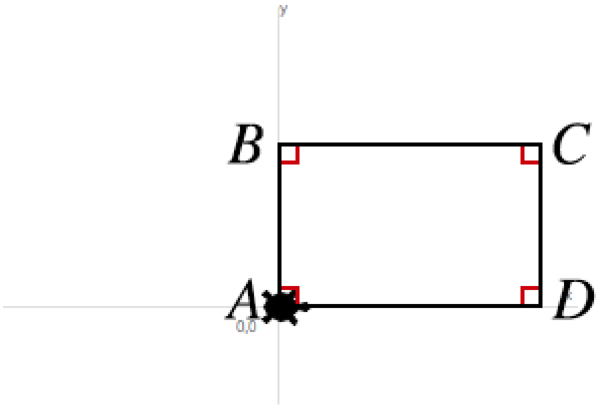

Figure 4

The distance at which the turtles moves is measured in pixels, that is, `turtle.forward(75)` means that it will move `75` pixels forward. **A pixel is the smallest part of an image**. All that we see on the monitor consists of the smallest pixels that together give a picture (Figure 5).

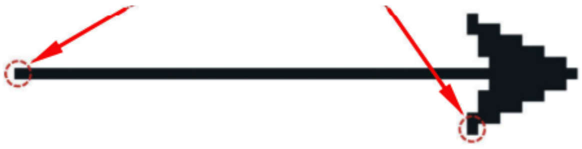

Figure 5

The movement of the turtle is specified using the functions `turtle.forward()` and `turtle.back()`, where we indicate distance in pixels in the parentheses, as was already mentioned.

The turtle drew a straight line but now it needs to turn around. We use the `turtle.left()` or `turtle.right()` function to specify direction of the turn (left or right). Indicate rotation in degrees in parentheses. **Degrees are a measure of angle and can be a value from 0° to 360°**. It is customary to call the angle equal to 90° right, 180° straight, 360° full angle (Figure 6).

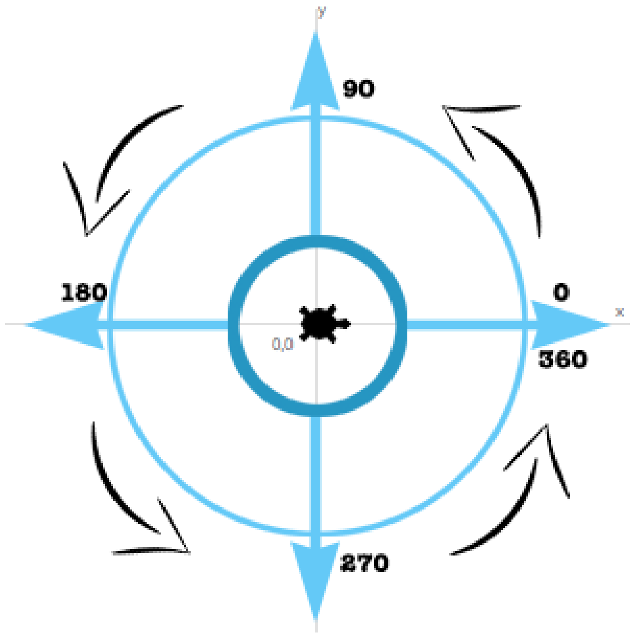

Figure 6

Let's try to make the turtle turn right by simply changing all the lines from `turtle.left(90)` to `turtle.right(90)` in the code.

So, you learned how to draw a simple figure using **Python** and the **turtle** library.

Drawing a Circle

In Python, you can use the turtle to draw a circle (see Figure 7 on page 11). To do this, you need only one function, `turtle.circle()`.

Let's do this with the help of this code:

```python
import turtle

window = turtle.Screen()
turtle.reset()

turtle.shape("turtle")
turtle.bgcolor("white")
turtle.color("red")

turtle.speed(2)
turtle.pensize(3)

turtle.circle(80)
turtle.penup()
turtle.forward(80)
turtle.pendown()

turtle.circle(80)
turtle.penup()
turtle.forward(80)
turtle.pendown()
turtle.circle(80)

window.exitonclick()
```

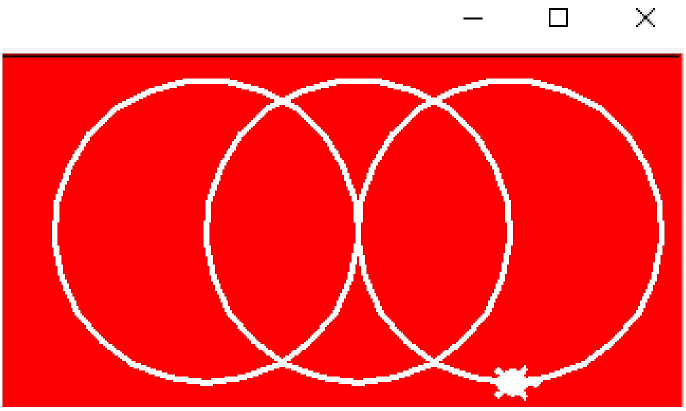

Figure 7

Here we use the function `turtle.circle(80)`, where `80` is the circle's radius (`R`), i. e. the segment connecting the middle of the circle (`O`) and any point that lies on the circle (Figure 8).

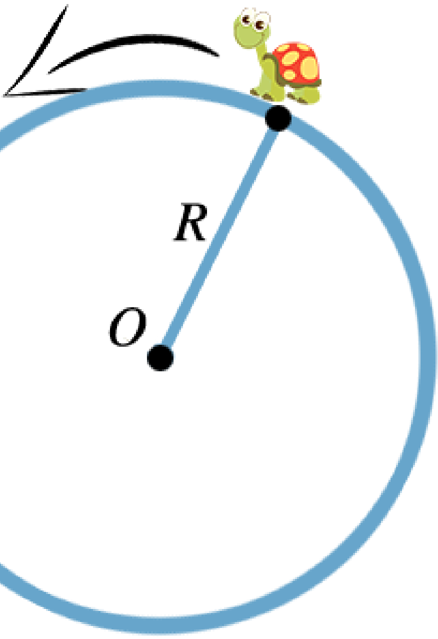

Figure 8

We used `turtle.penup()` and `turtle.pendown()`, let's look at how this works.

Our turtle is a pen, so as soon as we lift the pen, it won’t leave a track any more. Thus, we can move the turtle to any place without leaving a trace.

Comment out the lines that contain the commands *`#turtle.penup()`* and *`#turtle.pendown()`* and run the code.

Lesson # 2

Python Turtle Graphics. Simple Figures